**---trigger for updating sales and total\_sales from customer\_orders**

CREATE OR REPLACE FUNCTION public.update\_sales\_and\_totals\_simple()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

DECLARE

total\_sales\_sum NUMERIC(10, 2); -- Variable to store the total sales sum

BEGIN

-- For INSERT operation or UPDATE operation when order status is 'Completed'

IF (TG\_OP = 'INSERT' AND NEW.status = 'Completed') OR (TG\_OP = 'UPDATE' AND NEW.status = 'Completed') THEN

-- Insert or update the sales table

INSERT INTO sales (sale\_date, total\_amount, payment\_method, customer\_name, restaurant\_id)

VALUES (NEW.order\_date, NEW.total\_amount, NEW.payment\_method, NEW.customer\_name, NEW.restaurant\_id)

ON CONFLICT (sale\_date, customer\_name, restaurant\_id)

DO UPDATE

SET total\_amount = EXCLUDED.total\_amount; -- If conflict, update total\_amount

-- Calculate the total sales for the restaurant after the update

SELECT COALESCE(SUM(total\_amount), 0) INTO total\_sales\_sum -- Use COALESCE to handle NULLs

FROM sales

WHERE restaurant\_id = NEW.restaurant\_id;

-- Update the restaurant's total\_sales value only when order status is 'Completed'

UPDATE restaurant

SET total\_sales = total\_sales\_sum

WHERE restaurant\_id = NEW.restaurant\_id;

-- For DELETE operation on customer\_orders

ELSIF TG\_OP = 'DELETE' THEN

-- Delete the corresponding sales record

DELETE FROM sales

WHERE sale\_date = OLD.order\_date

AND customer\_name = OLD.customer\_name

AND restaurant\_id = OLD.restaurant\_id;

-- Recalculate and update the total\_sales in the restaurant table after the delete

SELECT COALESCE(SUM(total\_amount), 0) INTO total\_sales\_sum

FROM sales

WHERE restaurant\_id = OLD.restaurant\_id;

UPDATE restaurant

SET total\_sales = total\_sales\_sum

WHERE restaurant\_id = OLD.restaurant\_id;

END IF;

RETURN NEW; -- Always return NEW, as we're handling both INSERT/UPDATE and DELETE

END;

$BODY$;

ALTER FUNCTION public.update\_sales\_and\_totals\_simple()

OWNER TO postgres;

CREATE TRIGGER update\_sales\_and\_totals\_simple\_trigger

AFTER INSERT OR UPDATE OR DELETE ON customer\_orders

FOR EACH ROW

EXECUTE FUNCTION public.update\_sales\_and\_totals\_simple();

----**trigger for updating menuitem\_id based on section**

CREATE OR REPLACE FUNCTION set\_menuitem\_id()

RETURNS TRIGGER AS $$

BEGIN

-- Create a composite menuitem\_id by concatenating section\_id and a sequence number

NEW.menuitem\_id := CONCAT(NEW.section\_id, '-', LPAD(nextval('menuitem\_id\_seq')::TEXT, 3, '0'));

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER set\_menuitem\_id\_trigger

BEFORE INSERT ON menu\_items

FOR EACH ROW

EXECUTE FUNCTION set\_menuitem\_id();

----**trigger for generating unique\_id for staff**

CREATE OR REPLACE FUNCTION generate\_unique\_staff\_id()

RETURNS TRIGGER AS $$

BEGIN

-- Generate a custom ID in the format "STF-{Sequence Value}"

NEW.unique\_id := 'STF-' || LPAD(NEXTVAL('staff\_unique\_id\_seq')::TEXT, 6, '0');

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_generate\_unique\_staff\_id

BEFORE INSERT ON staff

FOR EACH ROW

EXECUTE FUNCTION generate\_unique\_staff\_id();

**---checking if more hours are logged in**

CREATE OR REPLACE FUNCTION check\_hours\_limit()

RETURNS TRIGGER AS $$

DECLARE

total\_hours NUMERIC(10, 2);

BEGIN

-- Calculate total hours worked by the staff member for the current week

SELECT SUM(hours\_worked)

INTO total\_hours

FROM timesheet

WHERE staff\_id = NEW.staff\_id

AND EXTRACT(WEEK FROM date) = EXTRACT(WEEK FROM NEW.date); -- Ensure we sum for the same week

-- Check if the total hours worked exceed the assigned hours

IF total\_hours + NEW.hours\_worked > (SELECT hours\_perweek FROM staff WHERE staff\_id = NEW.staff\_id) THEN

RAISE EXCEPTION 'Staff member % has exceeded the assigned weekly hours.', NEW.staff\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_hours\_before\_insert\_or\_update

BEFORE INSERT OR UPDATE ON timesheet

FOR EACH ROW

EXECUTE FUNCTION check\_hours\_limit();

**--Trigger for updating sale\_items from order\_items**

CREATE OR REPLACE FUNCTION update\_sale\_items()

RETURNS trigger AS

$BODY$

DECLARE

v\_sale\_id INT; -- Changed variable name to avoid conflict with table column

BEGIN

-- Step 1: Retrieve sale\_id based on order\_id and customer\_name

SELECT s.sale\_id

INTO v\_sale\_id

FROM sales s

WHERE s.sale\_date = (SELECT co.order\_date

FROM customer\_orders co

WHERE co.order\_id = NEW.order\_id)

AND s.customer\_name = (SELECT co.customer\_name

FROM customer\_orders co

WHERE co.order\_id = NEW.order\_id)

AND s.restaurant\_id = (SELECT co.restaurant\_id

FROM customer\_orders co

WHERE co.order\_id = NEW.order\_id)

LIMIT 1;

-- Step 2: Insert or update the sale\_items table

INSERT INTO sale\_items (sale\_id, menuitem\_id, quantity, section\_id)

VALUES (

v\_sale\_id, -- Using the variable to avoid ambiguity

NEW.menuitem\_id,

NEW.quantity,

NEW.section\_id

)

ON CONFLICT (sale\_id, menuitem\_id)

DO UPDATE

SET quantity = EXCLUDED.quantity,

section\_id = EXCLUDED.section\_id;

RETURN NEW;

END;

$BODY$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg\_update\_sale\_items

AFTER INSERT OR DELETE OR UPDATE

ON public.order\_items

FOR EACH ROW

EXECUTE FUNCTION public.update\_sale\_items();

**---trigger for calculating sick leaves based on hours worked from timesheet table**

CREATE OR REPLACE FUNCTION public.calculate\_sick\_leave\_hours()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Calculate sick leave hours based on hours worked

IF NEW.hours\_worked >= 10 THEN

UPDATE staff

SET sick\_leave\_hours = sick\_leave\_hours + (NEW.hours\_worked / 10)

WHERE staff\_id = NEW.staff\_id;

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.calculate\_sick\_leave\_hours()

OWNER TO postgres;

CREATE OR REPLACE TRIGGER trg\_calculate\_sick\_leave\_hours

AFTER INSERT OR UPDATE

ON public.timesheet

FOR EACH ROW

WHEN (new.staff\_id IS NOT NULL)

EXECUTE FUNCTION public.calculate\_sick\_leave\_hours();